DOCUMENT RESUME

ED 378 965 IR 017 028

AUTHOR Brigham, Frederick J.; And Others
TITLE Hypermedia Supports for Student Learning.
INSTITUTION Council for Exceptional Children, Indianapolis. Indiana Federation.
PUB DATE 25 Feb 94
NOTE 28p.; Paper presented at the annual meeting of the Indiana Federation, Council For Exceptional Children (Indianapolis, IN, February 25, 1994).
PUB TYPE Guides - Classroom Use - Teaching Guides (For Teacher) (052) -- Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Audiovisual Aids; *Authoring Aids (Programming); Computer Software; Computer Uses in Education; Educational Media; Educational Strategies; Educational Technology; Elementary Secondary Education; Higher Education; *Hypermedia; *Integrated Learning Systems; Learning Disabilities; *Multimedia Instruction; *Multimedia Materials; Multisensory Learning; Student Projects; Teaching Methods
IDENTIFIERS HyperCard; HyperStudio

ABSTRACT

Recent developments in multimedia and hypermedia have simplified the creation of high-quality presentations so that they are becoming a viable instructional option for both teachers and students. One potential use of integrated media technology is in student-created programs. To demonstrate this point, novice student programmers, who used simple authoring tools and were trained using a modified form of instructional scaffolding, constructed integrated media programs to develop or adapt instructional materials appropriate for use with students with learning disabilities. An overview of simple hypermedia tools suitable for teacher and student presentation creation is presented, including images, sound and text. The following examples of instructional programs are provided: HyperCard; Hyperstudio; and SemNet Academic. All three examples include program descriptions, comments from student programmers, and suggestions for classroom use. An appendix includes a table of software sources and costs, and two figures illustrate sample content maps. (Contains 12 references.) (MAS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED 378 965

IR

Hypermedia Supports for Student Learning

Paper presented at the annual meeting of the
Indiana Federation, Council For Exceptional Children
Indianapolis, Indiana
February 25, 1994

Frederick J. Brigham

Paula L. Hendricks

Stephanie M. Kutcka

Esther E. Schuette

Valparaiso University
Valparaiso, IN 46383

Running head: Hypermedia Supports

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Frederick Brigham

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR017028

BEST COPY AVAILABLE

Abstract

Recent developments in multimedia and hypermedia have simplified the creation of high-quality presentations so that they are becoming a viable instructional option for both teachers and students. In this paper, we present an overview of simple hypermedia tools suitable for teacher and student authoring of presentations including images, sound, and text. Three inexpensive programs, Hypercard, Hyperstudio, and SemNet Academic, will be demonstrated through the use of college student-created instructional programs. The students who created the applications were enrolled in a special education instructional methods course. None of the students had previous experience in computer programming and no more than a one semester hour introduction to computers course. The quality of their work indicates that with appropriate support and equipment, multimedia presentations may be a viable instructional tool for even novice computer users. Comments and suggestions based on the experience of our student programmers are also discussed.

Hypermedia Supports for Student Learning

Hypermedia is a term used to refer to software and hardware configurations which allow video, graphics, and sound to be linked to text in important and meaningful ways (Hasselbring, Goin, & Wissick, 1989). Specifically, the links between such media and text can be designed elaboration and understanding of information. While some authors refer to such projects as multimedia presentation, the Cognition and Technology Group at Vanderbilt Learning Technology Center (CTGV) (1993) has suggested that integrated media is a more appropriate term because it "...reminds us that our goal is to integrate media to facilitate learning rather than simply multiplying the number of media available to learners. The implication for students with learning disabilities is that student learning will be a function of the quality and quantity of the instruction received more than any other characteristic. By providing thoughtfully selected, text-relevant examples (Levie & Lenz, 1982) and clear, well executed instructional routines (Mastropieri & Scruggs, 1987) integrated media programs may provide another promising alternative for instruction of students with learning disabilities.

One potential use of integrated media technology is in student-created programs. Students provided with text selections or concepts requiring organization, elaboration and illustration, might be able to use simple authoring tools to demonstrate their knowledge and understanding. Also

integrated media programs may be used to provide enriched context and to anchor instruction to meaningful and concrete references (CTGV, 1993). We believe that teachers who have experience designing and operating such programs will be better able to assist their students in learning with integrated media than teachers who lack such experience. Based on this belief, we set out to construct examples of integrated media programs with novice programmers using simple authoring tools. Our purpose was not to develop sophisticated finished products, but to develop prototypical examples of user-authored programs.

Authoring Tools in Integrated Media

We chose three authoring tools to work with for our projects. Authoring tools typically involve trading a wider set of programming options called forth by complicated and abstract programming languages for a more limited set of programming options called forth by more straightforward commands involving verbal labels or menu selections. The authoring systems we worked with were (a) HyperCard, (b) HyperStudio and (c) SemNet Academic. Hypercard and HyperStudio are general purpose authoring systems which present text, graphics, video and sound information according to program design and user commands. SemNet Academic is a program dedicated to producing semantic maps (Johnson & Pearson, 1984) on the computer screen. Each authoring system is described in more detail along with the program developed for each system in later sections of the paper.

One of the major concerns for any program developer is with collecting hardware and software resources which are adequate for the demands of the program. In the next section, we describe the equipment used for the project, supporting software, and the instruction in the operation and use of the equipment and software provided to our student programmers.

Equipment and Instruction

In this section, we describe the equipment, programs and procedures we used to create our programs. There are a number of excellent examples of hardware and software applications available to the program author. We used equipment available to us at the university where the programs were written.

Equipment and Supporting Software

Computers. All of the programs presented today were constructed on a Macintosh IIci computer with 8 megabytes RAM and an 80 megabyte hard drive. A Quadra 950 with 16 megabytes RAM installed with a "VideoSpigot" card was used to produce "QuickTime" movies and to produce the images of still photographs used in the various programs. The images were transfered to the Macintosh IIci. Both computers were running under Apple System 7.1.

While we were able to construct these programs with an 8 megabyte computer, certain programs and images were quite constrained by the lack of memory. We note that the programs we used will run on machines with 2 or 4 megabytes of memory; however, it is our recommendation that 8 megabytes be

considered a theoretical lower limit for computer users wishing to construct integrated media programs. The Macintosh IIci was housed in the office of the first author, a faculty member in the University's Department of Education. The Quadra computer and video equipment were housed in the University's Academic Computing Center, in another building on campus.

Other hardware. In addition to the computers used for this project, we used a VideoSpigot card, a full page color scanner, and a MacRecorder sound utility. The VideoSpigot card transforms signals from camcorders, VCRs, and television sources into digital code which may be stored and replayed by computers. The full page scanner we used works much the same way as a photocopier. A static image is placed a glass plate in the scanner. This image is then exposed to a strong light source and reflected onto a light-sensitive screen. This image is then converted to a digital map and sent to the computer for processing and storage. MacRecorder is a device which transforms analog audio signals into digital information which can then be stored and processed by the computer.

Software utilities. Four software programs were employed in support of the authoring systems used to develop the actual instructional programs. These programs included: (a) SoundEdit Pro, (b) Adobe Photoshop, (c) Adobe Premiere and (d) Aldus Super Paint. SoundEdit Pro is bundled with MacRecorder and is a digital sound processor and editor.

We used SoundEdit Pro to remove some of the background noise from one of the audio sequences through a technique

known as compression. We also were able to select audio segments with greater precision by creating a visual map of the sound wave and cutting unnecessary information using the visual map as a reference to the temporal progression of the sounds. This technique saved a considerable amount of time compared to more traditional search and play forms of sound editing.

Adobe PhotoShop was used to add and correct colors on scanned images. Also various photographic effects such as filtering and blurring were examined but are not represented in the illustrations in out program. One feature available in the PhotoShop program, a cut/paste/extrapolate missing information routine proved to be very helpful in the creation of the images we used. In one case a picture of a frog also contained written text which covered part of the target image. By using this routine we were able to cut the unwanted information and reconstruct the parts of the image which were lost or hidden. PhotoShop can be a highly memory intensive application, depending on the type of image processed. The larger Quadra machine handled high-quality color photos well, but the smaller IIci was unable to deal with these images. Consequently, Photoshop was used to degrade the quality of images we used thereby allowing machines with smaller memory capacities to display the images.

We used Adobe Premier to edit the video clips which were captured by the Quadra and VideoSpigot card. Though the program contains a number of powerful features, we only used

the edit routines to select segments from larger video clips. One problem we encountered was the size of the files we created. Our desire was to show longer rather than shorter clips. Long clips easily exceeded the storage space on a floppy disk, even using high density disks. We therefore became very selective about the information that would be shown in dynamic video format as compared to static photographs.

Some of the images used in our programs were originally line drawings. We used Aldus Super Paint to touch up and enhance the scanned images of these drawings. Also, Super Paint has a scaling function which allowed us to create images which were all of roughly the same size for use on a computer-generated map.

Instruction and Support for Student Programmers

The student program authors were all education majors in their senior year and enrolled in a special education instructional methods class focussing on the needs of students with Learning Disabilities. Three of the student programmers were seeking an endorsement in special education, one was an elementary education major taking the course to enhance her instructional repertoire. The programs constructed were part of an assignment to develop or adapt instructional materials appropriate for use with students with learning disabilities. None of the student programmers possessed training beyond a one-semester hour orientation to computers in education before beginning their projects.

Scaffolded programming. Rather than engage in a formal instructional program to teach each authoring system to each student programmer, we used a modified form of instructional scaffolding (Vygotsky, 1978). Instructional scaffolding supposes that individuals may be supported at a higher level of performance under guidance than which they might function independently. In our project, the special education instructor served as the primary support for the student programmers by providing access to the equipment, an orientation to the project and basic instruction in the operation of the program. Thus, each student programmer received an initial overview of the authoring system they were to use including a set of basic commands specific to their program. With this background, the authors began to identify the content to be contained in their program and design rough formats for their screen layout. The course instructor was available to assist when the programmers encountered difficulty. At various times during the development of these programs, the instructor's knowledge was insufficient unto the demands of the program, particularly in creating and editing video segments. In this case, the instructor and student programmers visited consultants and advanced users on campus for additional training and guidance. Through this approach, we believe that each of us learned quite a bit of new information and computer techniques.

Examples of Instructional Programs

This section will describe the software packages used in our project and examples of student-developed programs employing

each package. A list of the software packages used, sources and approximate prices is provided in Table 1.

HyperCard "Elaborative Maps" Stack

Hypercard is perhaps the most simple of the authoring systems included in our presentation. It is adaptable from and entry-level programming utility to a highly sophisticated authoring environment. Hypercard is also available on most Macintosh computers already in classrooms. Few sophisticated programming techniques are required and most commands reflect common language, making this an attractive choice for the novice programmer.

Elaborative Maps. The application we developed with HyperCard presents a map of major battles of the American Revolution, each map symbol is keyed so that a click of the mouse brings up a mnemonic keyword and a voice describing the battle, its keyword, and the events associated with the battle. The maps were adapted from Brigham, Scruggs, and Mastropieri (in preparation) Elaborative Maps for enhanced learning of historical information: Uniting spatial, verbal, and imaginal information.

The "Elaborative Maps" study examined spatial displays of information. In general, Brigham, Scruggs, and Mastropieri found that the effectiveness of spatial displays in promoting recall of locations of unfamiliar places could be greatly increased by mnemonic encoding of the location's verbal labels compared to more traditional map location symbols. For example, the mnemonic keyword for Ticonderoga was tiger. The location of Ticonderoga was represented on the map by a picture of a tiger.

However, recall of associated information was promoted only by elaborative encoding. For example, the information "At the battle of Ticonderoga, they captured cannons to use later." was represented by a tiger (keyword) shooting a cannon (elaboration). It is presently unknown whether the spatial organization promoted by "hard-copy" maps will be obtained when a reduced image is presented on a computer screen and individual keyword elaborations are keyed to the map locations. This stack may be used in a future study to address this question.

Hyperstudio "Animals of the Rain Forest" Stack

Hypercard 2.1 is presently limited to black and white presentations though version 2.2 which is advertised as color-compatible is presently available. HyperCard also possesses many sophisticated options that may not be needed by teacher and student-programmers. If these advanced features are ignored, there is no problem for the novice programmer; however, novice users who call up advanced features may become frustrated and confused with the program. For example, HyperCard contains a powerful feature called Script Language. The power of script language is only available to programmers familiar with the commands and techniques. An alternative to Hypercard is Roger Wagner Software's authoring system, Hyperstudio.

Hyperstudio exploits the multi-media control capabilities of the microcomputer by adding color, animation, ability to control CD ROM drives and Videodisc players, as well as the basic

features of HyperCard. It is a menu-driven authoring tool. Menu-driven tools require little programming knowledge, supporting the user through a limited set of plain language commands. In such systems, the user often trades some level of power and sophistication for convenience and ease of operation.

The Animals stack. The application we created with Hyperstudio was designed to teach young elementary students about selected animals of the rain forest. An animation showing a coral snake crawling over the rough vegetation on the forest floor to help shed its skin was created from a videotape owned by the university using a Macintosh Quadra equiped with a "Video Spigot" card. A spider monkey is shown in its tree-top habitat and heard giving its high-pitched cry of alarm. The spider monkey image was scanned into the computer from a color photograph. The photograph was then transformed to black and white in order to conserve memory as color photographs are memory-intense components of such presentations. The sound of the monkey was imported from a videotape using the "MacRecorder" utility. Finally an examination of the bright colors used as a defense by the tree frog and its characteristic sound were presented. Black and white and color versions of the frog are presented from scanned images. A sound segment created with MacRecorder is also available for the user to hear.

User responses and records. Each segment contained a focus question to reinforce the major point of information regarding the animal. Each question contained a remedial loop for incorrect responses and appropriate feedback for correct

responses. An additional feature in Hyperstudio allows the computer to record the selections made for each question and number of times the user executed a particular loop or set of screens.

Comments from student programmers. The first problem we encountered was in finding a consultant experienced in the Hyperstudio program itself. Hyperstudio appears to be quite popular in educational settings but university-based computer or technology consultants appear to be more informed about programs designed for business and industry then education. Our advice to other student programmers is don't expect too much help from general consultants in regard to educational software.

The second difficulty that we encountered was in the scope of our project. Our original plan was designed to demonstrate a great number of the options available in the software package. While this plan would have yielded an impressive project, it proved to be to much for us to accomplish in the time we had avaliable to us. We therefore recommend that student programmers attempt to limit the scope of their project to a few well-designed sequences which are manageable rather than a large and elaborate program which is difficult to complete.

The major problem we encountered was with available memory in our computer. Although the eight megabyte machine which we used to develop our program is sufficient to most tasks we employed, large and complicated color images would sometimes cause the computer to run out of memory and

"freeze-up." Also, large color images slow the program because of increased computing time. We therefore suggest that programmers use smaller rather than larger color images in their programs. Also, use black and white photographs and line drawings whenever possible to conserve memory and speed program execution.

Student programmers' suggestions for classroom use. Based on the experience described in the preceding sections, we make the following recommendations for classroom use of Hyperstudio:

1. If possible, conduct a step-by-step demonstration in a laboratory situation to have each student create a prototype stack. This stack can then be used as a model for individual projects.
2. Create and display a summary poster or reference card of the most commonly used commands to cut down on time spent searching for them in the manual.
3. Have the student programmers set a specific goal or purpose for their stack. It may be helpful to consider the stack in terms of a goal and each individual card as an objective when planning the program.
4. Limit student programmer groups to two or three students.
5. Teach cooperative learning skills as a part of the programming experience.
6. The better the plan or storyboard, the better the project and the more likely it will be completed. Do not short change planning time for explorations on the computer!

Semnet Academic: Mapping Narration and Exposition

The final program created in our project demonstrated the uses and limitations of a semantic mapping program called SemNet Academic (Fisher, 1993). SemNet Academic is a scaled down version of a more elaborate concept-mapping program known simply as SemNet. SemNet incorporates basic elements of semantic mapping theory, allowing users to: (a) specify concepts, (b) describe relationships between concepts, and (c) identify details and concepts related to a superordinate concept. Additional features such as blocks of text and graphics may be attached to concepts or details. All of these features are then presented in a spatial display known as a concept map or semantic map. Concept maps typically take the form of arc-node networks with the relationship identified in the arc and the concept contained in a node (Holley & Dansereau, 1984). Such techniques have been suggested as appropriate strategies to support the performance of students with learning disabilities, thought the evidence regarding the effectiveness of these strategies is mixed (Brigham, 1992).

The concept map is displayed on the computer screen in what is referred to as "frames." Each frame contains a central concept (in an oval-shaped marker) connected to subordinate concepts or details (in rectangular markers) by a directional or bidirectional ray. For example John and Robert Kennedy would be connected by bidirectional rays because each is the brother of the other but Joseph Sr. and John may connected by directional rays indicating Joseph is the father of John and John is the son of

Joseph, respectively. SemNet Academic can only display one frame of a network at a time. Each frame of the concept map may be printed. The software stamps each printout with the date and time of the printout as well as the title of the map. Printouts of the main screen for each of the concept maps demonstrated today are included in Figures 1 and 2.

Narrative and expository maps. Two semantic networks were created with this program, one of a narrative text and one of an expository text. We selected these types of texts because we wanted to examine the appropriateness of semantic mapping with software in each type of text. The narrative text selected was Snow White, a familiar fairy tale. The expository text selected was a chapter regarding stimulant medications as a treatment for behavior problems (Poling, Gadow, & Cleary, 1991).

We were unable to adequately describe the sequence of events in the fairy tale using the semantic mapping software. This is probably because semantic mapping employs a parallel presentation of information whereas narrative texts rely on a linear progression of events. It is possible that narrative texts could be adequately described using this system, but in our experience this would not be easily done. It should be noted that the authors of Semnet Academic do not suggest this program for narrative texts. It is clear that such elements of comprehension such as character study and elaboration can be supported using this program; however; simple narrative formats may be better served by more traditional outlines.

Our implementation of SemNet Academic with the exposition of stimulant medication was quite different than the experience we had with expository texts. In elaboration, parallel presentation may be an advantage because student may gain a sense of the overall pattern of relationships characterizing a central concept. Additionally, we found the semantic map produced for stimulant medication contained a great deal of information in a compact and well-integrated package. This appears to be a use to which teachers and students could easily apply this program.

Comments from student programmers. We found this program to be quite user-friendly. Once the initial set of commands were mastered (and they were mastered fairly easily), the program itself was very straight-forward. It is sometimes difficult, however, to think in terms of the directional relationships specified by this program. Because of this, we believe that teachers should help their students discriminate between learning tasks in which this type of processing would be beneficial and those tasks which might benefit from another strategy. Our finding is that expository texts are easily represented in this software, but narratives are much more difficult to adequately portray.

Student programmers' suggestions for classroom use. Based on the experience described in the preceding sections, we make the following recommendations for classroom use of Semnet Academic:

1. This program strikes us as being better suited for individual than for group use. Knowledge may be organized in an idiosyncratic nature and the individual execution of the program would support that idiosyncrasy.
2. Before using this program teacher may wish to provide explicit instruction in the types of relationships employed in the program (reciprocal and bidirectional).
3. Have a plan of organization so that you can avoid inserting too much detail. Too much detail results in a cluttered map which may not be helpful to the learner.
4. In narrative texts, this program can be employed to create character sketches. Avoid temporally-related ideas.

Discussion

Our experience leads us to suggest that embedding the training on computers in another course is adequate to develop some basic level of competence but insufficient to actually promote higher levels of expertise and independence in use of computers to develop one's own instructional programs. While our project suggests that the approach we employed is unlikely to develop high levels of expertise, the student programmers involved in the project report that they feel more prepared to make wise use of technology consultants and resources in the schools where they will be employed than they did before engaging in this project.

A limit to the approach we employed is that the student programmers really worked with only one program. While the use of one program can lead to greater depth of understanding,

it does not necessarily promote breadth of understanding. One the other hand, working with limited resources is more of a given than an unusual circumstance for public school teachers. An experience which provides wide access to software and hardware which may not be available to teachers in real schools will probably be unrealistic and, therefore, unproductive.

On a positive note, student programmers were in agreement that working with a specific piece of authoring software was likely to make them more critical consumers of educational technology. One of the programmers remarked: "Anything you hear about technology in education sounds great at a conceptual level, but actually working with the technology encourages one to think more critically about its use in teaching and learning. Working through an instructional problem with a given technology forces one to move beyond the claims and the bells and whistles to the more important question, 'just what can this thing really do?" In the end, clear-headed answers to the questions "what can this thing do?" and "Is that the best way to approach my instructional problems?" are needed when considering application of instructional technology to the problems presented by real students in real schools. We believe that educators who have had a first-hand opportunity to explore answers to these questions through active development of their own solutions will be likely to make better decisions

Conclusion

With limited instruction in computer operation and program developers, college students enrolled in a special

education methods class were able to develop prototype instructional programs of sufficient quality to be used with school-aged children. The form of scaffolded programming which we employed appeared to be adequate to enable student programmers to build their own programs with comparatively little programming experience or instruction in programming techniques. This form of learning about computers was able to be incorporated into an existing class in special education teacher-training.

Like most computer application development projects, it should be noted that these projects represent a great deal of time and effort on the part of their authors. If our suspicions regarding the benefits of program authoring to promote teacher's abilities to assist their students in effectively using integrated media are correct, it will have been time well spent.

References

Brigham, F.J., Scruggs, T.E., & Mastropieri, M.A. (1994). Elaborative maps for enhanced learning of historical information: Uniting spatial, verbal and imaginal information. Manuscript submitted for publication.

Brigham, F. J. (1992). Spatial learning and instruction of students with learning disabilities. In T. E. Scruggs & M. A. Mastropieri (Eds.) Advances in Learning and Behavioral Disabilities (pp. 57-85). Greenwich, CT.: JAI Press.

Fisher, K. (1993). SemNet Academic. [Computer program]. Santa Barbara, CA: Intellimation.

Poling, A., Gadow, K.D., & Cleary, J. (1991). Drug therapy for behavior disorders: An introduction. Elmsford, NY: Pergamon Press.

Hasselbring, T.S., Goin, L.I. & Wissick, C. (1989). Making knowledge meaningful: applications of hypermedia. Journal of Special Education Technology, 10, 61-72.

Holley, C.D. & Dansereau, D.F. (1984). Networking: The technique and the empirical evidence. In C.D. Holley & D.F. Dansereau (Eds.) Spatial learning strategies: Techniques, applications and related issues (pp. 81-108). Orlando, FL: Academic Press.

Johnson, D.D. & Pearson, P.D. (1984). Teaching reading vocabulary (2nd ed.). New York: Holt, Rinehart & Winston.

Levie, W.H. & Lenz, R. (1982). Effects of text illustrations: A review of research. Educational Communication and Technology Journal, 30, 195-232.

Mastropieri, M.A. & Scruggs, T.E. (1987). Effective instruction for special education. Austin, TX: PRO-ED.

O'Keefe, M. (1988-1993). HyperStudio. [Computer program]. El Cajon, CA: Roger Wagner Publishing.

The Cognition and Technology Group at Vanderbilt Learning Technology Center (1993). Integrated media: Toward a theoretical framework for utilizing their potential. Journal of Special Education Technology, 12, 71-85.

Vygotsky, L.S. (1978). Mind in society. Cambridge, MA: Harvard University Press.

Table 1
Software Sources and Costs

Software Demonstrated/Described

Authoring Systems

Program	Publisher Source	Cost*
HyperCard 2.1 (2.2 now available)	Apple Computer MacWarehouse & other catalog outlets	$110.00 (list: 150)
HyperStudio	Roger Wagner Publishing 1050 Pioneer Way, Suite P.; El Cajon, CA 92020	$100.00
Semnet Academic	Intellimation 130 Cremona Drive P.O. Box 1922, Santa Barbara, CA 93116-1922	$65.00

Utilities

Adobe Premiere 3.0**	Adobe Sysyems, Inc. MacWarehouse & other catalog outlets	$459.00 (list: 695)
Adobe Photoshop 2.5**	Adobe Systems, Inc. MacWarehouse & other catalog outlets	$549.00 (list: 895)
Sound Edit Pro	MacroMind Paracomp MacWarehouse & other catalog outlets	$200.00 (list: 295)
Super Paint	Aldus MacWarehouse & other catalog outlets	$99.00 (list: 150)
QuickTime Starter Kit	Apple Computer MacWarehouse & other catalog outlets	$109.00

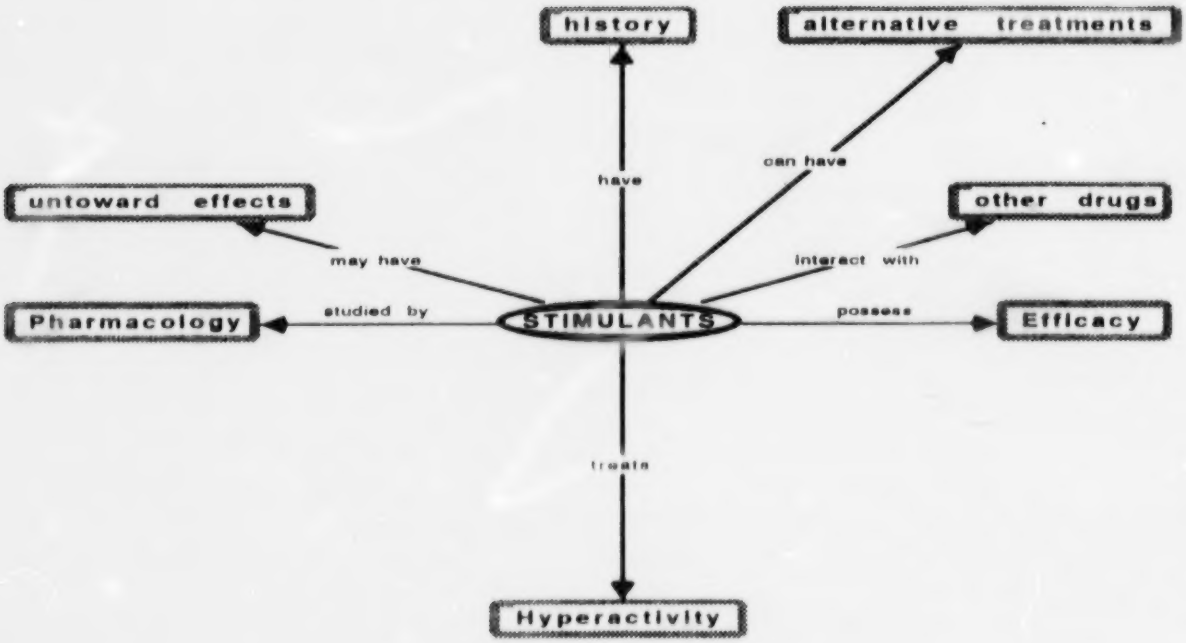
* Figures provided are approximate as of January, 1994.

** A package version of Photoshop and Premiere entitled Adobe Audition is available in the range of $150.00 to $250.00. Audition contains limited versions of Photoshop and Premiere.

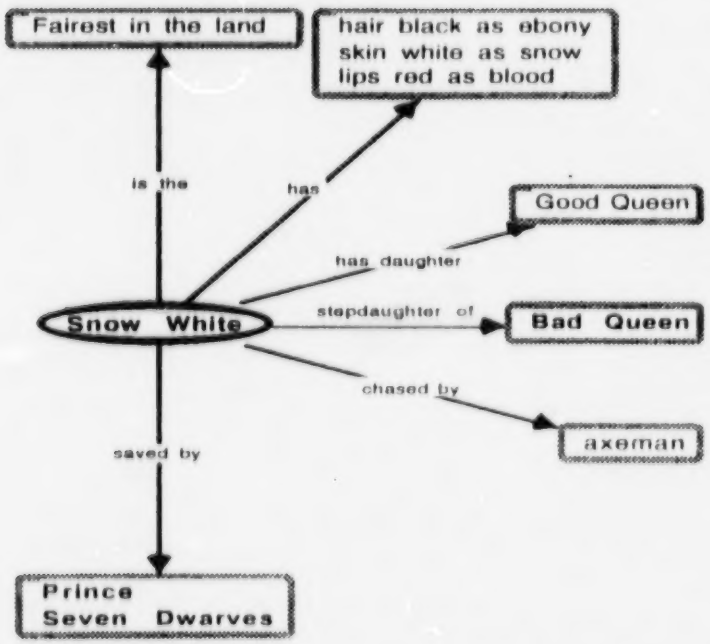
Figures 1 and 2

Sample Content Maps

2/24/94 10:49 AM Stimulants



2/24/94 10:50 AM Snow white 2



BEST COPY AVAILABLE